一种固定选图数的实时幸运成像算法

王锦良[1]，李彬华[1,2]

（1. 昆明理工大学 信息工程与自动化学院，云南 昆明 650500）

（2. 昆明理工大学 云南省计算机应用技术重点实验室，云南 昆明 650500）

摘 要：幸运成像技术是一种基于大量短曝光图像中选取少量幸运好图进行配准、叠加的高分辨率图像恢复技术，能够有效减小大气湍流对图像质量的影响，但传统的基于中央处理器（CPU）的幸运成像算法难以实现实时化。本文利用现场可编程逻辑阵列（FPGA）的并行性和灵活性优势，提出了一种新的基于 FPGA 的幸运成像算法并构建了一个 FPGA 实验系统。该算法采用一种固定选图数且无需排序的图像选择策略和一种以行列坐标为基准的图像配准策略，能够有效地节省算法处理时间和硬件资源，达到实时幸运成像的目的。这个新的实时幸运成像算法能够以简洁的方式在中小规模的 FPGA 上实现，所得高分辨率图像与基于传统 CPU 算法处理的结果完全相同。实验表明，对于 2000 帧 128×128 像素的输入图像进行幸运成像处理，本算法的运行速度比本实验室之前提出的算法快 27 倍，比传统的基于 CPU+MATLAB 幸运成像算法速度快 150 多倍，处理帧率可达 197 帧/秒。该算法及其 FPGA 实现技术可以用于构建真正实时的幸运成像系统。

关键词：图像处理；大气湍流；实时性；幸运成像；FPGA

中图分类号：TP391.4　　**文献标识码:A**　　**文章编号:**

幸运成像技术是一种用于去除大气湍流效应的简单有效的图像恢复技术[1]。由于该方法需要从大量的短曝光图像中选取少量幸运图像进行配准和叠加，计算量大，通常是在天文观测完成之后的一段时间内用专门的计算机进行图像恢复或重建，难以实时处理。不过实时化仍是其发展方向之一[2]。对于大量图像的幸运重建来说，传统的中央处理器（Central Processing Unit，CPU）的个人计算机（Personal Computer，PC）无能为力。因此，近年来具有强大并行处理能力的图形处理器（Graphics Processing Unit，GPU）和现场可编程逻辑阵列（Field Programmable Gate Array，FPGA）器件作为硬件加速器引入图像处理领域[3-5]。虽然 GPU 具有强大的并行处理能力，但其实现的系统体积较为庞大，处理速度上与 FPGA 相比并没有任何优势，这一结论是 Jackson 分别在 GPU 和 FPGA 上实现幸运区域融合（Lucky-region Fusion，LRF）算法（其基本思想也源于短曝光幸运图像）的实验中得出的[6]。

2008 年西班牙的 Oscoz 和 Piqueras 所在的团队在 FastCam 项目研究过程中，首次将幸运成像技术在 FPGA 硬件上实现，并获得了实时的高分辨率图像[7,8]。FastCam 是基于实时阈值进行选图的，虽然实时性有保证，但可能导致所选图像出现偏差，特别是由于 FastCam 系统没有剔除包含宇宙射线的图像，当输入宇宙射线图像后，会选中这帧图像并使实时阈值增大，导致后续的好图不再被选中。这些不足使得该系统的处理能力和成像效果受到一定限制。而且他们的论文中仅介绍了算法基本框架和实验结果，并没有介绍各个处理模块的具体设计方法和实现技术。

2018 年本实验室赵盼孜等人首次在论文中对幸运成像算法的 FPGA 实现进行了初步细致的描述[9]。虽然赵的系统在功能上是完善的，但由于在图像传输链路采用的是数据速率较低的 SD 卡接口，同时幸运成像算法与基于 CPU 的算法是一致的，虽然各模块功能和时序划分比较独立，但综合的资源利用率和时钟效率较低，导致该实验系统的处理能力和速度都没有达到实时化的要求，性能上与 FastCam 系统有较大差距。

基金项目：国家自然科学基金（11673009）资助.

收稿日期：

作者简介：王锦良，男，硕士研究生. 研究方向：成像技术与图像处理. E-mail：410784005@qq.com

通讯作者：李彬华，男，教授. 研究方向：成像技术与图像处理、天文探测器技术. E-mail：lbh@bao.ac.cn

作者新近的论文介绍一种新的实时幸运成像算法及其实现技术[10]，其中采用的滤波、宇宙射线检测与剔除、选图、配准、叠加等子算法与 FastCam 和赵盼孜的算法完全不同。但由于采用固定选图比的选图方法，所选出的好图与传统方法稍有差异，导致幸运成像结果也有微小的偏差。

本文以本实验室赵的幸运成像系统为基础，采用千兆网作为图像输入通道，提出了一种具有实时处理能力的基于 FPGA 的幸运成像算法，该算法创新性地采用一种固定选图数且无排序的图像选择策略和一种以行列坐标为基准的图像配准策略，能够有效地节省算法处理时间和硬件资源；该算法在好图的配准和叠加等环节与文献[10]的方法也有差异。然后，在一块中小规模的 FPGA 开发板上实现了这一算法，并构建了一个紧凑的实时化的高速幸运成像系统，能够实现大量图像的幸运重建。最后，利用云南天文台丽江观测站 2.4m 天文望远镜拍摄的大量目标短曝光图像进行 FPGA 算法设计的可行性验证，并对系统的性能进行了分析讨论。

1　实时幸运成像算法的 FPGA 实现

由于研究工作的延续性，本文提出的幸运成像算法的硬件实现平台和设计开发软件环境与赵盼孜完成的系统相同，即以 Xilinx 公司 Spartan-6 系列的 XC6SLX16 芯片为核心的开发板和一台 CPU 为 Xeon E5620 的工作站，设计开发环境是 ISE Design Suite 14.7，采用 ChipScope pro 14.7 和 ModelSim SE 10.5 进行硬件设计的验证与调试。在本文的研究过程中，首先复现了赵系统幸运成像算法的 FPGA 实现过程，然后与本系统幸运成像算法的 FPGA 实现过程作对比。

经典的幸运成像算法包括短曝光图像的预处理、选图、配准、叠加四个算法流程[3]，用 FPGA 实现幸运成像算法，也应该有这几个算法流程。本文基于实时处理的要求，以新的选图、配准、叠加思想设计了相应的算法及其实现模块。

1.1 算法的总体设计

1.1.1 算法概述

一个设计合理的基于 FPGA 的实时算法，基本思想与传统的基于 CPU 的算法是一致的，但必须摒弃传统的串行处理的习惯思维，尽可能多地采用并行处理思想进行设计和实现。所以，在设计新的基于 FPGA 的实时幸运成像算法，我们仍然采用传统的基于 PC 或 GPU 的幸运成像算法[1,3]的关键思想——选图、配准和叠加，但数据处理方式与步骤则按并行思想进行。首先将图像数据分成两路，一路进行图像的高斯滤波及数据暂存，一路是宇宙射线检测和选图、配准；然后并行叠加，得到高分辨率图像；最后，采用锐化操作对高分辨率图像进行增强，并送往视频图形阵列（Video Graphics Array，VGA）显示。算法的关键步骤描述如下：

（1）图像滤波，同时并行地查找图像峰值；
（2）保存滤波后图像，同时并行地检测剔除宇宙射线；
（3）按固定的选图数进行选图；
（4）按图像序号和行列坐标进行配准；
（5）并行叠加；
（6）对叠加图像做反转变换进行增强同时保存重建图像；
（7）送往 VGA 显示。

需要说明的是，由于 FPGA 具有高度的并行处理能力，不仅是上述（1）、（2）两步中是并行处理的，后面的那些步骤，也可以跟图像输入、滤波等过程同时工作。这是基于 FPGA 的处理算法与基于 CPU 的幸运成像算法的最大不同点。

传统的基于 CPU 的幸运成像算法是一种事后图像处理方法，是在所有图像采集并传输

完成后，按照固定的选图比选取固定数目的好图。因此，对于传统的幸运成像，固定选图比与固定选图数的实质是一样的。但对于实时处理而言，需要在图像采集和传输的同时进行图像选取，按照固定选图比的方式会丢失少量的好图[10]，而在一定传输帧数下按照固定选图数的方式则不会丢失任何一帧好图。从上述算法流程可见，本算法选图是按选固定的选图数目进行的。虽然这既与传统的采用固定选图比的选图方案[1,3,9]不同，也与采用固定选图比的动态选图方案[10]不同，但当输入图像达到设定值时，所选出的幸运图像（即好图）与传统的采用固定选图比选出的好图结果是一致的。由于采用固定的选图数目，也没有选择阈值，所以有别于 FastCam 中的实时阈值选图方案。这是本算法的一个创新点。以上算法中各步骤的详细处理过程，构成相应的一些子算法，它们与 FPGA 的结构相关，将在后面相关小节中介绍。

1.1.2 FPGA 系统的总体设计

FPGA 是一个半定制硬件，基于这种器件的算法设计，与传统的基于 CPU 的算法设计差别巨大，因为 FPGA 算法的实现与硬件的结构、资源的种类和大小等密切相关。对于上面所述的幸运成像算法，主要的步骤可用一个 FPGA 模块电路去实现，再结合图像输入输出方式，其实现的系统主要包括千兆以太网接收模块、高斯滤波模块、宇宙射线剔除模块、数据输入缓存模块、数据读写控制与存储模块、短曝光图像的选图、配准、叠加模块、图像灰度变换模块、重建图像保存模块以及 VGA 驱动显示模块，系统的总体结构框图如图 1 所示。其中千兆网、滤波和射线剔除这三个模块是赵的系统中没有的，选图、配准和叠加这三个模块重新进行了设计，灰度变换模块采用反转变换这种简单易实现的锐化操作对高分辨率图像进行增强，其他模块做了少量的优化。本系统 DDR3 工作频率为 312.5MHz（注意，所提出算法不限于 312.5MHz，系统也可以在 625MHz 下正常工作）。

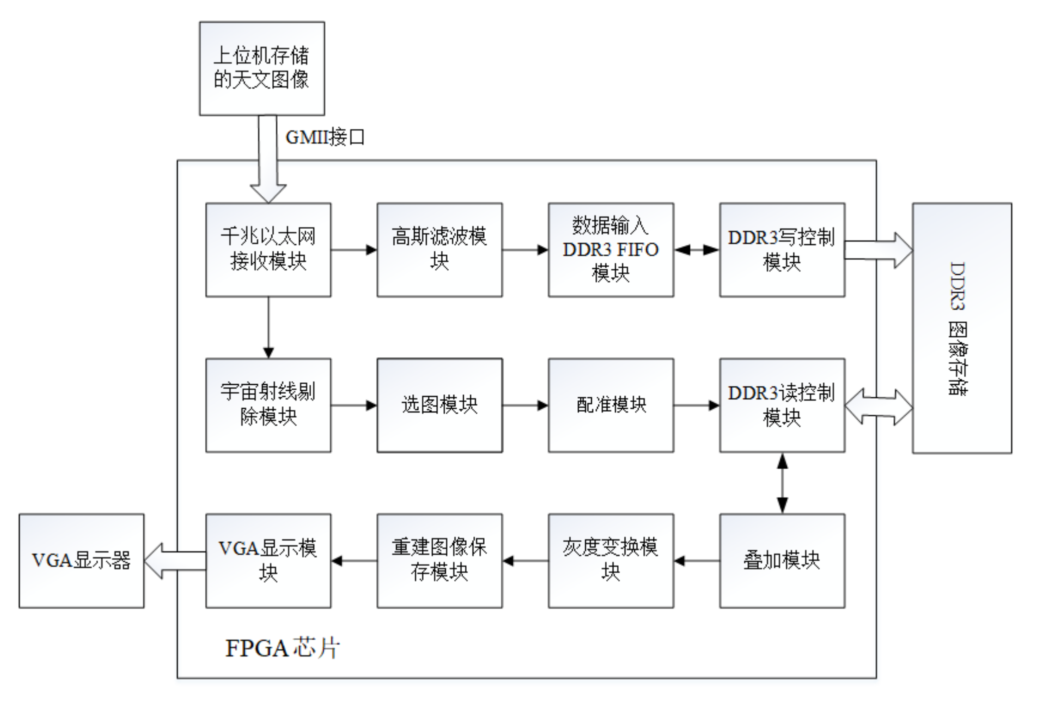


图 1 实时幸运成像算法的总体结构框图

Fig. 1 Overall structure of the real-time lucky imaging system.

由于受到所用 FPGA 芯片 RAM 资源以及硬件逻辑资源的限制，本系统目前的目标与 FastCam 一致，可以使用 2000 帧 128×128 像素大小的短曝光图像作为原始图像进行选图处理，在配准过程中截取 64×64 像素大小的图片进行配准处理，并最终以 64×64 像素大小显示幸运成像结果。

由于千兆以太网接收模块的代码是由 FPGA 开发板厂商提供的代码修改得出的，其他与赵系统类似的模块可参阅赵盼孜的论文(文献[9])，这些模块本文不作介绍，高斯滤波模块与宇宙射线剔除模块可参阅文献[10]，本文也不作介绍。以下主要介绍新的思想方法重新设计选图、配准和叠加这三个算法及其实现模块。

1.2 选图算法及其实现模块设计

传统的选图思想是基于每帧图像的瞬时斯特列尔比进行比较排序。当图像较多时，选图算法计算量巨大，消耗的资源也很多，不适合在中小规模的 FPGA 系统上实现。为此，本文采用一种固定选图数目且无需排序的选图思路，符合 FPGA 的逻辑设计，可节省大量的时钟资源与硬件资源，该算法示意图如图 2 所示。

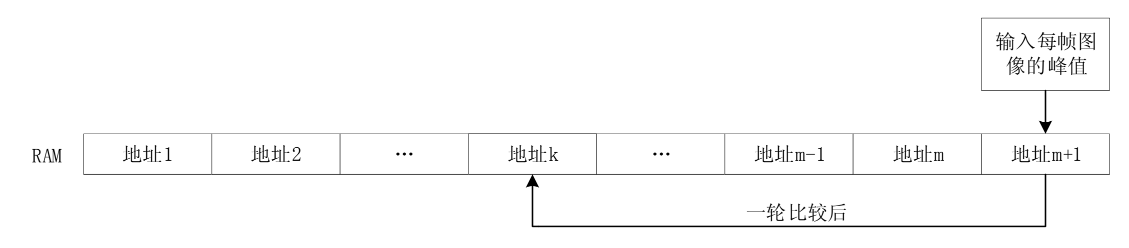


图 2 选图算法示意图

Fig. 2 Schematic diagram of the image selection algorithm.

图 2 中 m 代表固定的选图数目，也就是从 n 帧天文图像中选出 m 个峰值进行配准叠加，本系统实现的是从 2000 帧天文图像中按 1%的选图比[3,7]选出 20 个峰值进行配准叠加，故本系统的 m=20，n=2000。每来一个经过宇宙射线剔除模块后的峰值都暂存到 RAM 地址 m+1 上，然后依次读取 RAM 地址 1 到地址 m+1，相互比较并记录最小值所在的 RAM 地址 k，一轮比较结束后，把暂存在地址 m+1 上的峰值写入到最小值所在存储地址单元 k 中，如此往复，直到 n 帧图像峰值读取完毕后，RAM 中地址 1 到 m 中数据就是从 n 个图像峰值中的前 m 个最大值，依次读出，选图结束。

本文选图思想与文献[10]是不同的：本文采用的是固定选图数目的方法，即 m 是不变的；而文献[10]采用的是固定选图比的方法，即 m 是变化的。相应地，FPGA 实现的方法和 Verilog HDL 代码也大不同。文献[10]的优点是可以随图像的输入动态选图，但代码设计复杂，代码量大，特别是其选图结果与传统的算法有小的偏差[10]。本文算法的优点是在确定的输入图像数时选图结果与传统的算法完全一致，即不会丢失任何一帧好图，代码设计简单。

1.3 配准算法及其实现模块设计

本模块设计的目的是对经过选图模块选出的 128×128 像素大小的图像，以峰值所在像素为图像中心，截取其所在行列以及其前 31 行和后 32 行、前 31 列和后 32 列，形成一帧 64×64 像素大小的图像进行叠加处理。设计中的主要问题是，如何找到以峰值所在像素为中心的 64×64 像素大小图像的第一个像素在 DDR3 存储器中的地址，用以从 DDR3 存储器中读取相应的图像像素的灰度值，送入叠加模块处理。本文根据 DDR3 存储器数据存储原理以及图像配准要求，采用峰值所在图像序号和峰值所在像素位置（即横纵坐标）进行图像配准，得到了符合 FPGA 逻辑实现的配准公式，如式(1)所示。

$$addr_r = (pic_cnt - 1) \times 65536 + [(axis_x - 32) \times 128 + axis_y - 32] \times 4 \qquad (1)$$

其中，addr_r 为以峰值所在像素为中心的 64×64 像素大小图像的第一个像素在 DDR3 存储器中的地址，pic_cnt 为信息寄存器中寄存的图像序号，axis_x 和 axis_y 分别为信息寄存器中寄存的峰值所在像素的横坐标和纵坐标。公式(1)中乘 4 是因为一个像素在 DDR3 存储器中占 4 个地址单元，65536(128×128×4)代表 1 帧图像在 DDR3 存储器中占用的地址单元的个数。需要注意的是，如果一帧图像中存在两个或多个相同的峰值时，本系统取第一个峰值为中心

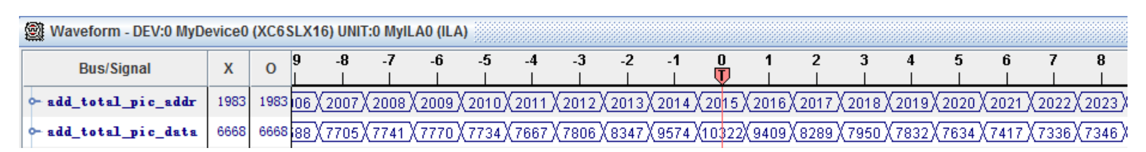
进行配准。

另外，由于本文中好图在 DDR3 中的存取方法与文献[10]的方法不同，以上的配准公式(1)与文献[10]的配准公式也就有明显的区别。

1.4 叠加算法及其实现模块的设计

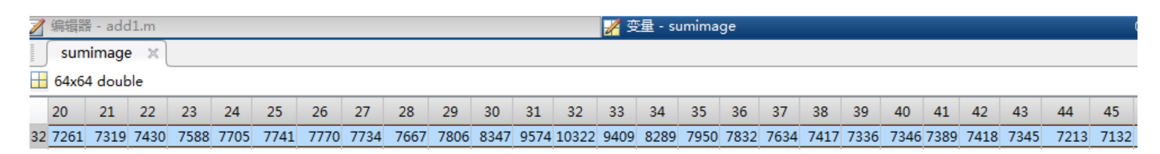
本模块实现的功能是将配准之后的图像进行叠加，将叠加后的图像送入灰度变换模块，为了实现图像数据的无缝缓冲和处理任务，并节省缓冲区空间，本系统采用一个 16 位真双口 RAM 和一个 16 位简单双口 RAM 进行乒乓操作实现图像的叠加。乒乓操作主要是依靠切换两个 RAM 之间的读写操作实现的，即当 RAM1 进行写操作时，RAM2 进行读操作，当一帧图像叠加完毕后，切换状态，即 RAM1 进行读操作，RAM2 进行写操作，直到 m 帧图像叠加完毕，此时 RAM2 中存储的就是全部叠加后的图像。

为了验证本模块设计的正确性，本系统采用固定 20 帧选图（即 2000 帧×1%）在 FPGA 上做预处理、选图、配准及叠加操作，并用 ChipScope 软件对最终结果图中峰值周围部分像素值进行抓取，抓取结果如图 3(a)所示，并在 MATLAB 上进行同样的处理，截取结果如图 3(b)所示。



(a) FPGA 处理的结果

(a) Results of FPGA processing



(b) MATLAB 处理的结果

(b) Results of MATLAB processing

图 3 叠加模块结果

Fig. 3 Superposition module processing results

对比图 3(a)和图 3(b)可知，经 FPGA 实现的结果与在 MATLAB 上实现的结果完全一样，说明叠加模块设计正确，同时也验证了本文所提出的幸运成像算法的各主要模块及其互联接口的设计是正确的。

3 实验结果及其分析

为了验证本文所提出的基于 FPGA 的幸运成像算法的性能以及 FPGA 实现的正确性，首先用同样的算法在基于 PC 的 MATLAB 处理平台上对天文目标的短曝光图像进行处理，其次在基于 FPGA 开发板的硬件处理平台上进行同样的处理，最后将相关结果做对比分析。

实验中所用天文目标短曝光系列图像是 2016 年 10 月 20 日在云南天文台丽江观测站对天文双星 HDS 70 的实测图像，共 10000 帧，这组图像的观测条件和参数参见文献[3]。在图像帧数以及图像大小的选取上，由于所用的现场可编程门阵列的限制，多次随机从 10000 帧图像中抽取 2000 帧，并裁剪成 128×128 像素大小的天文目标短曝光图像进行处理。

3.1 幸运成像算法在 CPU+MATLAB 平台上的处理实验

本实验的硬件平台是：Dell Precision T5500 图像工作站，其中内存 16GB，CPU 为 Xeon E5620，显卡为 NVIDIA GTX1080，运行的软件环境是 Windows 7 操作系统、MATLAB R2017a。

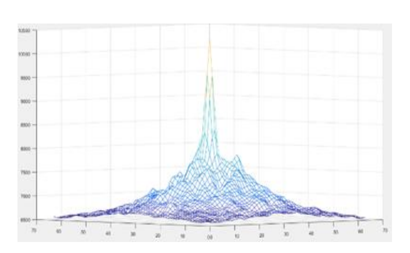
在 MATLAB 中对随机抽取的 2000 帧128×128像素大小的图像进行幸运成像算法处理，

在实际处理中只截取64 × 64像素大小的目标区域。根据文献[3,7]的研究结果，选图比取为1%，幸运成像效果最好。也就是从 2000 帧图像中选 20 帧做叠加，所得结果如图 4 所示。对图 4(a)的高分辨率图像进一步分析，采用二维修正矩算法[11]对目标（主星和伴星）进行计算，得到以像素为单位的主星坐标（33.015，32.147）和伴星坐标（37.906，26.068），由此计算出两星之间的距离约为 7.802 像素，再根据望远镜成像系统的比例尺 0.043 角秒/像素[3]，最终得到双星间距为 7.802×0.043≈0.335 角秒。这个结果与 HDS 70 在星表中的距离（0.3 角秒）[12]差别不大，误差主要来源于双星在星表历元位置与我们实际观测时刻的双星位置之间的差异和定心计算误差。这说明用 MATLAB 实现的幸运成像算法是正确的。



(a) 二维灰度图

(a) 2D gray image



(b) 三维灰度分布图

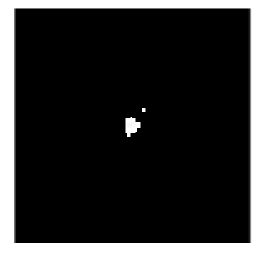
(b) 3D gray distribution map

图 4　2000 帧短曝光图像中选 20 帧时所得的幸运成像结果

Fig. 4 Results of 20 frames of lucky images selected from 2000 frames of short exposure

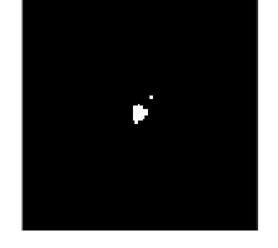
3.2 幸运成像算法在 FPGA 平台上的实验验证

本实验的硬件平台是 Xilinx 公司 Spartan-6 系列的 XC6SLX16 芯片为核心的开发板。本系统基于 FPGA 上实现的幸运成像算法设计的验证，实时输入 2000 帧原始图像，采用采用固定 20 帧选图（相当于 1%的选图比例）做处理。由于验证所用图片总帧数以及选图比例过低，导致最终得到的高分辨率图像像素灰度值过低，所以无论在 MATLAB 上还是 FPGA 上做算法处理所得的高分辨率图像都比较模糊，故本系统在 FPGA 和 MATLAB 上均对算法处理所得的高分辨率图像做了相同的灰度变换处理——锐化，将最终得到的高分辨率图像中感兴趣的目标或区域突出出来，分别得到了如图 5(a)、5(b)所示的锐化图像。



(a) FPGA 处理的结果

(a) Result of FPGA processing



(b) MATLAB 处理的结果

(b) Result of MATLAB processing

图 5 基于 FPGA 和 MATLAB 实现的幸运成像算法处理所得的结果图像

Fig. 5 Resultant images Obtained by lucky imaging algorithm based on the FPGA and the MATLAB

图 5 (a)是通过以太网回传最终高分辨率图像锐化结果（数据）给 PC 机并调用 MATLAB 的函数显示的结果图，图 5(b)是在 MATLAB 上进行幸运成像算法处理并做相同锐化的结果图。通过对比可以看出两帧图像是完全相同的，从而验证了本系统在 FPGA 上实现的幸运成像算法设计正确。

3.3 对比实验及其结果分析

为了体现本文所提出的基于固定选图数的幸运成像算法的优越性，本文设计了二个对比实验。第一个对比实验：分别用本文所提出的基于固定选图数的算法、赵盼孜提出的基于排

序的算法[9]、文献[10]中提出的基于固定选图比的算法以及在 MATLAB 上实现的相同算法对 1000 帧原始图像按 1%的选图比例进行幸运成像实验，就它们的算法运行时间、图像数据读取时间和总运行时间进行对比，实验结果如表 1 所示。注：由于赵的系统处理能力有限，这里只对 1000 帧图像进行处理。

表 1 不同计算平台下 1000 帧原始图像的幸运成像算法运行时间（单位：秒）

Tab. 1 Running times (in seconds)

计算平台	算法运行时间	图像数据读取时间	总运行时间
本文固定选图数算法	0.09	4.99	5.08
赵盼孜排序算法[9]	2.45	41.94	44.39
固定选图比算法[10]	0.08	4.99	5.07
CPU+MATLAB	13.65	0.15	13.80

第二个对比实验：对同帧数同大小的原始图像在本文所提出的基于固定选图数的幸运成像算法实现、FastCam 项目[8]中基于实时阈值的幸运成像算法实现以及文献[10]中基于固定选图比的幸运成像算法实现资源消耗进行对比。注：这里选 2048 帧图像进行处理，是为了与西班牙 FastCam FPGA 系统做对比。

表 2 不同计算平台下 2048 帧 128×128 像素大小的原始图像幸运成像算法资源消耗

Tab. 2 FPGA on-chip resource consumption of lucky imaging for 2048 frames of cropped images

计算平台	片资源（片）	块 RAM 资源（块）
本文固定选图数算法	1319	19
FastCam 实时阈值算法[8]	3778	133
固定选图比算法[10]	1350	21

表 1 和表 2 中，本系统算法的 FPGA 实现和第一代系统算法的 FPGA 实现的硬件平台均为 3.2 节所述的 FPGA 开发板，CPU+MATLAB 实现的硬件平台为 3.1 节所述的平台。

从表 1 可以看出，本文所提出的幸运成像算法用 FPGA 实现后的算法处理速度比赵的系统快 27 倍，同比提升了 96.33%，比 CPU+MATLAB 实现的算法处理速度快约 152 倍。虽然与固定选图比算法处理速度几无差异，但本文算法与传统算法结果完全一致，而固定选图比算法与传统算法结果稍有差异。

此外，从表 2 中可以看出，本文所提出的幸运成像算法用 FPGA 实现的片资源消耗仅为 FastCam 系统中片资源消耗的 34.91%，块 RAM 资源消耗仅为 14.28%，相比固定选图比算法实现的资源消耗也略有优势。

通过以上对比分析可知，本文所提出的幸运成像算法的 FPGA 实现在算法处理速度上和资源消耗上都具有一定的优势，图像帧率可达 197 帧/秒（约 5ms/帧）。天文幸运成像观测通常使用电子倍增电荷耦合器件（Electron Multiplying Charge-Coupled Devices，EMCCD）的相机，这是一种高速低噪声的科学级相机[13]。实际观测时，EMCCD 相机的曝光时间通常不会小于 20ms。例如，FastCam 系统的曝光时间是 30ms[7]，而我们在丽江 2.4m 望远镜上观测时的曝光时间是 20ms[3]。所以，本文提出的基于 FPGA 的幸运成像算法及其实现技术，可以用于构建实时的幸运成像系统。

4 结论

本文在仔细分析已有实时幸运成像算法和系统优缺点的基础上，充分利用 FPGA 的并行性与灵活性优势，提出了一个符合 FPGA 开发特点的实时幸运成像算法。该算法在选图方面，摒弃传统的排序选图思想，而采用固定选图数且无需排序的方式进行选图；在图像配准方面，采用好图的二维行列坐标进行配准，公式简洁明了。文中介绍了这一实时幸运成像算法的流程和各关键步骤所对应模块的逻辑设计方法和实现技术，以及模块的测试结果与验

证过程。以此为基础，在一块中小规模低成本的 FPGA 开发板上进行了系统的实现，构建了一个具有实时处理能力的紧凑的幸运成像系统。使新的幸运成像算法在中小规模的 FPGA 上可以处理 2000 帧以上的图像，达到或者说略超 FastCam 的处理能力，但成像结果与传统算法一致，资源消耗更少。由于处理图像（128×128 像素）的平均帧率可高达 197 帧/秒，所以，本文提出的幸运成像算法及其 FPGA 实现技术，可以用于构建实时的幸运成像系统。

致谢

中国科学院云南天文台张西亮、季凯帆、金振宇为本研究工作提供了原始天文图像，特此致谢。

参考文献:

[1] Law N M, Mackay C D, Baldwin J E. Lucky imaging: high angular resolution imaging in the visible from the ground[J].Astronomy and astrophysics,2006,446(2):739-745.

[2] Mackay C D. High-Efficiency Lucky Imaging[J], MNRAS, 2013, 432(1):702-710.

[3] 毛栊哗,李彬华,张西亮,季凯帆,金振宇．基于2m级大口径望远镜的幸运成像算法的实验研究[J]．光学技术, 2018,44(05):542-548.

[4] Robert Stewart, Kirsty Duncan, Greg Michaelson, Paulo Garcia, Deepayan Bhowmik, Andre w Wallace, RIPL: A Parallel Image Processing Language for FPGAs[J], ACM Transactions o n Reconfigurable Technology and Systems (TRETS), 2018, 11(1):7:1-7:24.

[5] 郭诚欣,陈红,孙辉,等．基于现代硬件的并行内存排序方法综述[J]．计算机学报，2017, 40(9):2070-2092.

[6] Jackson C R. Real time mitigation of atmospheric turbulence in long distance imaging using the lucky region fusion algorithm with FPGA and GPU hardware acceleration[D]. University of Delaware, 2015.

[7] Oscoz A, Rebolo R, Lopez R, et al. FastCam: a new lucky imaging instrument for medium-sized telescopes[C], Proc.of SPIE, 2008, Vol.7014:701447-1:701447-12.

[8] Piqueras J, Rodriguez-Ramos L, Martin Y, et al. FastCam: Real-Time Implementation of the Lucky Imaging Technique using FPGA[C]. Programmable Logic, 2008, Southern Conference on. IEEE, 2008:155-160.

[9] 赵盼孜,李彬华,毛栊哗,陶勇．基于现场可编程门阵列的幸运成像算法的实现[J]．天文研究与技术, 2019,16(02):236-243.

[10] Wang J, Li B, Xing K. A New Real-time Lucky Imaging Algorithm and Its Implementation Techniques, IEEE Access, 2020, 8(1): 52192-52208.

[11] 季凯帆,王锋. CCD图像的二维修正矩定心方法[J]．天文学报, 1996, 37(1):85-90.

[12] Stelle Doppie., "00317+1929 HDS 70", < http://stelledoppie.goaction.it/index2.php?iddoppia=2129 > (2010), http://stelledoppie.goaction.it/index2.php?iddoppia=2129.

[13] 胡泊，李彬华．低温下EMCCD电子倍增模型[J]．电子学报, 2013, 41(9):1826-1830.

A real-time lucky imaging algorithm with fixed number of selected images

Jinliang Wang[1], Binhua Li[1,2]

(1. Faculty of information engineering and automation, Kunming University of Science and

Technology, Kunming 650500, China)
(2. Key Laboratory of Applications of Computer Technologies of the Yunnan Province, Kunming University of Science and Technology, Kunming, 650500, China)

Abstract: Lucky imaging is a high-resolution image restoration technology by selecting, registering and superposing a small number of lucky images from a large number of short-exposure images. It can effectively reduce the effect of atmospheric turbulence on image quality. However, the traditional CPU-based lucky imaging algorithm is difficult to achieve real-time. Taking advantage of the parallelism and flexibility of FPGA, this paper proposes a novel FPGA-based lucky imaging algorithm and built an FPGA experimental system. The algorithm adopts an image selection strategy only selecting a fixed number of lucky images and without sorting, and an image registration strategy based on the row and column coordinates of the selected images, which can effectively save algorithm processing time and hardware resources, and achieve the purpose of real-time lucky imaging. The new real-time lucky imaging algorithm can be implemented on small and medium-sized FPGA in a simple way. The high resolution image obtained by the real-time lucky imaging algorithm is the same as that processed by the traditional CPU-based algorithm. Experiments show that, for a lucky imaging processing of 2000 frames of the input images of 128×128 pixels, the running speed of the FPGA-based algorithm is 27 times faster than that of the algorithm previously proposed by our laboratory, and more than 150 times faster than that of the traditional lucking imaging algorithm based on CPU+MATLAB, the processing frame rate is up to 197 fps. The proposed algorithm and its FPGA implementation technology can be used to build a true real-time lucky imaging system.

Key words: image processing; atmospheric turbulence; real-time; lucky imaging; field programmable gate array (FPGA)